Approximate convex hull of affine iterated function system attractors

Anton Mishkinis, Christian Gentil, Sandrine Lanquetin, Dmitry Sokolov

Approximate convex hull of affine iterated function system attractors

Anton Mishkinis[1], Christian Gentil[1] , Sandrine Lanquetin[1] and Dmitry Sokolov[2]
[1]LE2I - Université de Bourgogne
[2] LORIA - Université Nancy I

2011

Abstract

In this paper, we present an algorithm to construct an approximate convex hull of the attractors of an affine iterated function system (IFS). We construct a sequence of convex hull approximations for any required precision using the self-similarity property of the attractor in order to optimize calculations. Due to the affine properties of IFS transformations, the number of points considered in the construction is reduced. The time complexity of our algorithm is a *linear* function of the number of iterations and the number of points in the output convex hull. The number of iterations and the execution time increases logarithmically with increasing accuracy. In addition, we introduce a method to simplify the approximation of the convex hull without loss of accuracy.

Key words: iterated function system, IFS attractor, convex hull approximation

1 Introduction

Iterated function systems (IFS) define objects whose geometry can be very complex. This geometry is determined by a given set of transformations. An attractor may be evaluated by iterating these functions. Not only is this evaluation expensive, but the analysis and characterization (location, size) of the resulting shape can be complex. It is therefore interesting to have a more conventional form bounding the attractor.

The problem of bounding the IFS attractor occurs in many tasks, including numerical fractal analysis or the localization of an attractor. To guarantee the objects manufacturability, it is important to take into account the severe production constraints. So we must be able to quickly evaluate the approximation and localization of an attractor.

The approximate convex hull may also be used to estimate normal vectors at points of an IFS fractal for real time realistic visualization.

One of the most challenging tasks in computer games is fast and accurate collision detection. A typical game environment is modelled by a collection of triangle meshes representing the scene geometry. Usually complex objects with fractal structure consist of a large number of tri-

angles. Construction of the approximate convex hull will facilitate collision computations. In addition to accuracy, the approximate convex hull of various parts of the IFS attractor can be constructed.

In this paper, we demonstrate how the properties of an IFS may be exploited to compute convex hulls at any required accuracy. The article is organized in the following way: we start by recalling the basic concepts of an IFS and notations in section 3. In section 4, we examine each step of Martyn's approach [1] in order to generalize it to 3D and to optimize it. We show how to simplify the approximation of the convex hull, i.e., to reduce the number of points without losing accuracy. We then focus on the complexity of Martyn's approach and the complexity of our algorithm in section 5. Finally, in section 6, we compare results obtained with our algorithm and with Martyn's before concluding.

2 Related work

Methods to calculate approximations of the convex hull of an IFS attractor have already been developed.

Strichartz, Wang, Kenyon et al. [2, 3] studied the boundaries and the convex hulls of self-affine tiles that can be considered as the attractors of very special affine IFS, where all the transformations have the same linear part.

Lawlor and Hart [4] presented an algorithm to construct a tight bounding polyhedron of the IFS attractor. An algorithm expresses the IFS-bounding problem as a set of linear constraints on a linear objective function, which can then be solved via standard techniques for linear convex optimization. This method works for a predetermined number of convex hull faces and shows the interactive rate only when this number is small.

More recently, Duda [5] and Martyn [1] have presented methods to calculate the approximate convex hull of the affine IFS attractor. The two methods are similar. The former is based on the so-called "width function" that returns the nearest to the point bounding half-space in a given direction. The latter is based on constructing a sequence of balls that bound corresponding parts of an attractor to approximate the convex hull. The approach is presented in 2D only.

Another important problem in computing the convex hull is to determine a bounding ball for the attractor of an IFS.

Gentil [7] described an approach based on the dichotomous search for the minimal radius of the ball that bounds the attractor of an IFS. The approach can be applied in a multi-dimensional space and calculates the result for a given precision.

Hart and DeFanti [8] introduced a method which starts with the unit ball centered at the origin. The algorithm iteratively produces a sequence of balls converging to the limit ball that bounds the attractor.

Rice [9] improved on Hart and DeFanti's approach by optimizing the radius of the bounding ball with the aid of a generic optimization package. He also showed that the center of the limit ball can be determined analytically by solving a system of linear equations.

Martyn [10] showed that the solution of this system is the centroid of the attractor with particular weights. To obtain a better approximation, he presented a heuristic iterative method called "balancing the attractor". The algorithm is not limited by the dimension of the space in which the attractor lies.

More recently, Martyn [11] presented a novel approach to approximate the smallest disc to enclose an affine IFS attractor at any accuracy. The method is based on a concept of spanning points he introduced to describe the extent of an IFS attractor.

In this article, we study an approximation of the convex hull of a given affine IFS attractor. This approximation will be given in the polytope form. Our model can be considered as a generalization and an optimization of Martyn's method. Our algorithm constructs a sequence of convex hull approximations using the self-similarity property of the attractor in order to reduce the number of necessary operations. In

addition, we introduce a method to simplify the approximation of the convex hull without losing accuracy.

3 Background and notations

In this section we recall the major definitions and properties of iterated function systems as well as establish the notations used in this paper.

3.1 Iterated function system

Generally, an IFS is defined in a complete metric space $(\mathbb{X}, d)$, where d is the associated metric. The transformation $T : \mathbb{X} \to \mathbb{X}$ is called contracting if and only if there exists a real s, $0 \leqslant s < 1$ such that $d(T(x), T(y)) < s \cdot d(x, y)$ for all $x, y \in \mathbb{X}$. The minimal coefficient s which satisfies the inequality is called the contraction coefficient of the transformation T.

We are substantially interested in attractors that are produced by an IFS composed of affine transformations. Each transformation can be described as follows: $T_i : x \mapsto Lx + b$, where L is the linear part and b is the vector of translation.

Thus, an affine IFS $(\mathbb{X}, \{T_i\}_{i=0}^{N-1})$ consists of a finite set $\{T_0, \cdots, T_{N-1}\}$ of contracting affine transformations in a complete metric space $(\mathbb{X}, d)$. Let $\mathcal{H}(\mathbb{X})$ be the space of non-empty compact subsets of $\mathbb{X}$. Let d_H be the Hausdorff distance induced by the metric d, i.e.:

$$d_H(A, B) = \max\{d(A, B), d(B, A)\},$$

where

$$d(A, B) = \max_{a \in A} \min_{b \in B} d(a, b).$$

Then $(\mathcal{H}(\mathbb{X}), d_H)$ is a complete metric space. The Hutchinson operator $\mathbb{T} : \mathcal{H}(\mathbb{X}) \to \mathcal{H}(\mathbb{X})$ associated with the IFS is defined by:

$$\mathbb{T}(K) = \bigcup_{i=0}^{N-1} T_i(K)$$

If $s_{max} = \max_{i=0,\cdots,N-1} s_i < 1$ then $\mathbb{T}$ is also contracting in the complete metric space $(\mathcal{H}(\mathbb{X}), d_H)$. According to Banach fixed point theorem [12], $\mathbb{T}$ has a unique fixed point $\mathcal{A}$. This fixed point is named the IFS attractor, i.e.:

$$\mathcal{A} = \bigcup_{i=0}^{N-1} T_i(\mathcal{A}). \tag{1}$$

The attractor of an IFS may be evaluated recursively. That is, it can be approximated by a sequence of objects $\{P_n\}_{n \in \mathbb{N}}$, which converges to $\mathcal{A}$. An initial element in the sequence defines by means of a primitive $P \in \mathcal{H}(\mathbb{X})$. The following elements are defined recursively:

$$\begin{aligned} P_0 &= P \\ P_{n+1} &= \bigcup_{i=0}^{N-1} T_i(P_n) \end{aligned}$$

Elements P_n are the images of composite functions applied to P.

3.2 Approximate convex hull of an attractor

Let us consider an affine IFS $(\mathbb{X}, \{T_i\}_{i=0}^{N-1})$ in the complete metric space $(\mathbb{X}, d)$. The algorithm that we describe here constructs an approximating convex hull of an IFS attractor. For a given precision ε, we construct a sequence of convex hull approximations. The accuracy of each subsequent element increases. The output convex set will be an ε-approximation of the convex hull.

The ε-approximation C' of the convex hull of a non-empty compact $A \in \mathbb{X}$ is a set which satisfies the following condition:

$$d_H(C, C') \leqslant \varepsilon,$$

where C is the exact convex hull of A.

4 Calculating the approximate convex hull

To begin, we consider the algorithm developed by Martyn to calculate the approximate convex hull of an attractor in 2D. We will describe the

basic idea only in the following; details are available in [1]. We then explain how to compute the convex hull more efficiently by taking advantage of certain properties of affine transformations.

4.1 Martyn's method

Martyn's algorithm can be divided into four steps. In this section we describe them in more detail.

The first step consists in determining a bounding ball (not necessarily a tight one) with center c and radius r, denoted by $B(c, r)$. Thus each point of the attractor is situated at a distance which is less than r from the center c. By construction, the radius r satisfies the following inclusion:

$$\bigcup_{i=0}^{N-1} B(T_i(c), s_i r) \subset B(c, r).$$

According to definition (1) of the attractor of an IFS, we have: $\mathcal{A} = \bigcup_{i=0}^{N-1} T_i(\mathcal{A})$. So each ball $B(T_i(c), s_i r)$ approximates more precisely the corresponding part $T_i(\mathcal{A})$ of the attractor. We can therefore write the following inclusions:

$$\mathcal{A} \subseteq \bigcup_{i=0}^{N-1} B(T_i(c), s_i r) \subset B(c, r).$$

By applying all the transformations T_j for $j \in \{0, \cdots, N-1\}$ to the transformed centres $T_i(c)$, we obtain new balls such that:

$$\mathcal{A} \subseteq \bigcup_{i,j} B(T_j T_i(c), s_j s_i r) \subset \bigcup_{i=0}^{N-1} B(T_i(c), s_i r).$$

Thus, the union of these balls approximates $\mathcal{A}$ with smaller Hausdorff distance. We can iterate this construction to achieve the desired accuracy ε. The stopping criteria will thus be:

$$r * s_{\alpha_1} * ... * s_{\alpha_k} < \varepsilon, \tag{2}$$

where k is the number of iterations performed ($k \in \mathbb{N}$) and s_{α_i} is the contraction coefficient of the transformation T_{α_i}.

In this way we can construct the evaluation tree of an IFS, such that the product of contraction coefficients in each brunch is less than ε/r. In order to optimize the calculations, Martyn introduce the adaptive-cut method to determine the particular balls in this tree.

The second step consists in determining the lowest and the highest centres of the balls (with respect to their y-coordinates). Using the lowest point as initial, we then recursively determine the next convex hull vertex using the depth-first search that minimizes a polar angle with respect to the previous vertex as origin.

Iterating this method we compute the convex hull of the centres of all the balls obtained at the k^{th} iteration, i.e. we compute the ε-approximation of the attractor convex hull.

Finally, to calculate an ε-approximation covering the attractor, we have to offset E on ε.

4.2 Suggested modifications

Globally, our algorithm contains the same steps as Martyn's, but the approximate convex hull is determined at each level of the tree. The choice of this algorithm is guided by the following considerations. As the calculation of the convex hull is the most expensive step, we reduce this complexity by computing intermediate convex hulls and merging them. These intermediate hulls are calculated at each iteration to eliminate the unnecessary interior points. This allows us to reduce the number of points considered at the next iteration.

Moreover, at each iteration we perform a simplification of the approximate convex hull. The algorithm complexity is calculated in section 5.

4.2.1 Initialization

The first step of our algorithm is also to determine the bounding ball $B(c, r)$. We have chosen the method suggested in [7] because of its simplicity of implementation. In addition to efficiency, this approach allows to achieve better convergence as it computes the minimum bounding ball for a given accuracy.

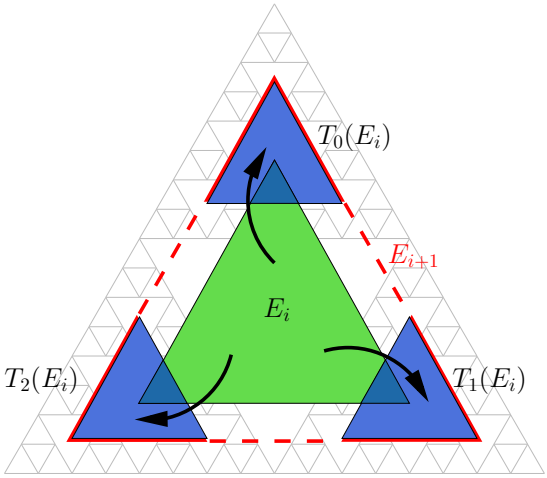


Figure 1: Illustration of the iterative construction of approximations. Given an ε_i-approximation of the convex hull, we calculate its images by applying all the IFS transformations to its vertices. We then merge all these images to obtain the ε_{i+1}-approximation of the convex hull.

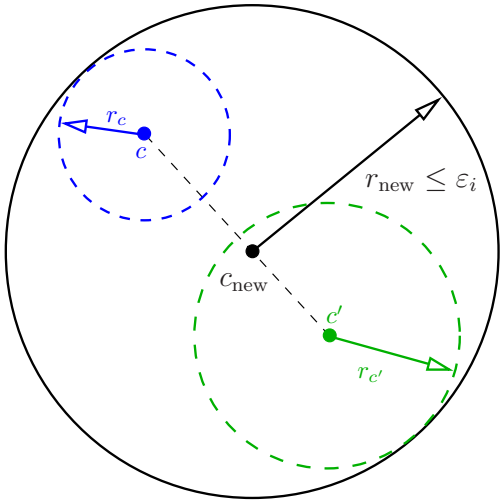


Figure 2: Illustration of the simplification process. If two neighboring points c and c' are close, i.e. $d(c, c') + r_c + r_{c'} \leqslant 2\varepsilon_i$ we can then replace them with the middle point without loss of accuracy.

4.2.2 Iterative algorithm

As mentioned above, the approximation of the convex hull is determined iteratively. This process is initialized by the convex hull of the singleton $\{c\}$. c is the center of the initial bounding ball which can be considered as an r-approximation of the convex hull.

Suppose that at the i^{th} step we have an ε_i-approximation of the convex hull, denoted by E_i. By construction, E_i is a convex polytope. We calculate the images of E_i by applying all the transformations T_j for $j \in \{0, \cdots, N-1\}$ to its vertices. We then merge all these images to obtain the ε_{i+1}-approximation of the convex hull. We can note that $\varepsilon_{i+1} \leqslant s_{max} \cdot \varepsilon_i$, where s_{max} is the maximal contraction coefficient of the transformations $T_0, \cdots, T_{N-1}$.

The process is repeated iteratively to achieve an ε-approximation of the convex hull. Figure 1 illustrates this iterative process.

4.2.3 Remarks

Removal of interior points. In the case of affine transformations, the following statement holds: *The image of a convex hull of points is a convex hull of the images of these points.* This means that the points inside the intermediate convex hull may be eliminated for the next iteration. In this way, we reduce the number of points considered in the calculations.

Stopping criterion. The stopping criterion, given by the inequality 2, is not optimal, because an IFS can contain transformations with different scale factors in different directions. By applying such transformations to the bounding ball, we obtain an ellipse covering the corresponding part of an attractor. By keeping track of the directions with the maximum and minimum scaling, it is therefore possible to increase the convergence rate to the exact convex hull.

For example, suppose an IFS with the two following transformations:

$$T_0 = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.9 \end{pmatrix}, \quad T_1 = \begin{pmatrix} 0.9 & 0 \\ 0 & 0.1 \end{pmatrix}.$$

After the application of T_0T_1 to the initial bounding ball $B(c, r)$, we obtain a ball which covers the corresponding part $T_0T_1\mathcal{A}$ of the attractor and approximates it with an accuracy of $0.81r$. Indeed, by construction, the radius of this ball is defined as follows: $r' = 0.9 \cdot 0.9 \cdot r$.

Let us consider a more precise approximation by ellipses. Suppose an IFS with the same transformations. The initial bounding ball $B(c, r)$ can be considered as an ellipse defined by the equation $X^T DX = 1$, where X is a coordinate vector $\begin{pmatrix} x \\ y \end{pmatrix}$ and D is the following matrix:

$$D = \begin{pmatrix} \frac{1}{r^2} & 0 \\ 0 & \frac{1}{r^2} \end{pmatrix}.$$

After the application of T_0T_1 to the initial bounding ball we obtain an ellipse for which the semi-major axis is determined by the following equation:

$$r' = \frac{1}{\sqrt{\rho}},$$

where ρ is the spectral radius of the matrix ${(T_0T_1)^{-1}}^T D(T_0T_1)^{-1}$, i.e. the absolute value of the dominant eigenvalue. Note that we can easily avoid extra computations and constraint on the reversibility of the applied IFS transformations by inverting this matrix. Indeed, a spectral radius ϱ of the inverted matrix $(T_0T_1)\, D^{-1} {(T_0T_1)}^T$ equals $\varrho = \frac{1}{\rho}$, therefore, the semi-major axis can also be determined by the following equation:

$$r' = \frac{1}{\sqrt{\rho}} = \sqrt{\varrho} = 0.09r.$$

Thus, the new ellipse approximates the corresponding part of the attractor with an accuracy of $0.09r$. This means that we increase the convergence rate if we calculate the dominant eigenvalue of this matrix at each iteration.

To calculate the spectral radius our original symmetric matrix can be reduced to tridiagonal form and then we could easily compute the dominant eigenvalue by QL method. So the absolute value of the dominant eigenvalue can be estimated in $O(dim^3)$, where dim is a dimension of the IFS transformation matrices.

Simplification of the convex hull approximation. We simplify the approximation of the convex hull at each iteration by replacing vertices that are close enough with one single vertex. To identify such vertices, we simply need to check the neighboring points on the approximating convex hull.

If two neighboring points c and c' satisfy:

$$d(c, c') + r_c + r_{c'} \leqslant 2\varepsilon_i \tag{3}$$

we can then replace them with the middle point without loss of accuracy, as shown in figure 2.

Thus, the number of points considered in the construction of intermediate convex hulls is reduced and the intermediate convex hulls are simplified without losing accuracy.

To avoid a constant rebuilding of data structures simplification can be performed during the merge.

4.2.4 Bounding approximation of the convex hull

The final step is offsetting the convex hull on ε to obtain the bounding approximate convex hull. As for the initialization, this step does not differ from that described in Martyn's algorithm, except that it will operate in a more dimensional space. We have chosen the method suggested in [13] because of its simplicity of implementation. This method offsets a mesh by moving all vertices along the multiple normal vectors of a vertex. Yoo [14] described a more accurate approach that uses distance fields based on the subdivision of the grid cells and an implicit surface interpolation to obtain an accurate model of arbitrary shape.

5 Complexity

In this section, we describe implementation details and compare the complexities of the algorithm developed by Martyn and our optimizations.

5.1 Convex hull of an attractor

First of all, as mentioned in section 4.1, Martyn's algorithm consists of four steps. The third one, i.e. the computation of the convex hull, is the most expensive step in the algorithm. More precisely, if an IFS consists of N transformations and we have to perform k iterations to obtain a given precision ε, the algorithm requires $O\left(h\sqrt{N^k}\right)$ time and $O\left(\log\left(N^k\right)\right)$ space, where h is the number of the approximate hull vertices. The number of iterations k can be found from the precision ε and the maximal contraction coefficient s_{max} using the following formula: $k = \lceil \frac{\log \varepsilon / r}{\log s_{max}} \rceil$.

5.2 Our method

The method developed by Martyn to calculate the approximate convex hull of an attractor is expensive in computation but requires little memory. This is because all the calculations are made without storing many intermediate points. Our optimization, featuring the elimination of the interior points, represents a complete renewal of the method.

The salient idea of our method is to take a convex hull with h_i points and to apply all the transformations of the IFS one time to generate $h_i N$ points. The convex hull of these points is then constructed and the interior points are eliminated. Construction of the new convex hull can be effected by using Chan's method [15] in $O\left(h_i N \log h_{i+1}\right)$. However, to avoid extra dependence on h we construct it by using direct merging. Application of N transformations generates N convex hulls, each of size h_i. The merging of two convex hulls is linear on the number of points in the convex hulls. So, the direct merging of N convex hulls requires $O\left(h_i N^2\right)$ time.

The total time complexity of our algorithm is $O\left(hN^2k\right)$ and the space complexity is thus $O\left(\max\limits_{i=1\ldots k} h_i\right)$. Note that for a given IFS, the number of IFS transformations N is constant, so the complexity of our algorithm is *linear* on the number of iterations k. This is a qualitative gain compared to Martyn's algorithm which depends *exponentially* on k. Also, our algorithm is output sensitive, since it depends on the number of approximate convex hull vertices h.

The space complexity of our approach is inferior to that required by the method developed by Martyn. Note that the number of iterations k grows logarithmically with the reduction of ε, so the optimized version of the algorithm requires logarithmic time to increase accuracy.

Moreover, in many cases we are likely to perform fewer iterations than Martyn's approach, since working with ellipsoids increases the convergence rate. The price we must pay in this case is increasing the complexity by a factor of dim^3 (that is, from $O\left(hN^2k\right)$ to $O\left(hN^2k \cdot dim^3\right)$, where dim is a dimension of the space, in which IFS is defined. However, dim does not depend on the precision ε, so it is a constant for a given IFS. Thus, our algorithm is still linear on the number of iterations k.

The following pseudocode illustrates the implementation of our algorithm:

```
Input: An affine IFS (𝕏, {T_i}_{i=0}^{N-1}) and an accuracy ε
Output: An ε-approximation of the convex hull
initialBall ← Calculate initial ball B(c, r) for a given IFS
// Initial approximation of the convex hull containing
// a single point c with an accuracy r
conv ← new Polytope(initialBall)
while(maximum accuracy of conv vertices > ε)
{
    // Create a variable for a new convex hull
    newConv ← Empty
    for each transform T_i of the IFS
    {
        // Calculate the image of the convex hull
        E ← T_i(conv)
        // Merge the image with newConv
        if (newConv = Empty)
            newConv ← E
```

```
        else
            // Simplify the convex hull during a merge
            newConv ← Merge(newConv, E)
    }
    conv ← newConv
}
return conv
```

The merging of convex hulls in 3D can be effected by approach described in [15]. However, because of its complexity of implementation, we simply recalculate the convex hull of all hN intermediate convex hull vertices at each iteration. So, the time complexity of the 3D results, presented in section 6, is actually $O\left(hNk \cdot \log(hN) \cdot dim^3\right)$.

6 Results and discussion

We develop an application to test the method presented in this paper. Our application was designed and coded in C# language.

We examined the presented algorithm for various IFS attractors and different precisions in 2D and 3D spaces. It should be pointed out that all tests were performed on an ordinary PC (Intel®Core^{TM}2 Duo T7500 2.2GHz 3GB RAM) without any additional computational facilities.

Several results for various IFS attractors are presented in figures 3 and 4. For each attractor, the program generates an approximation of the convex hull at the specified accuracy. The execution time of our algorithm is presented here:

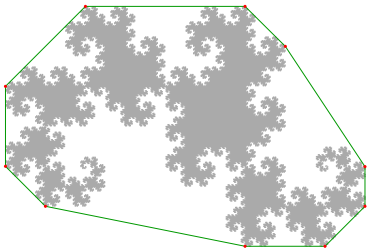
Attractor	N	dim	ε	time(ms)
Sierpinski triangle	3	2	10^{-9}	16
Koch curve	4	2	10^{-9}	27
Dragon curve	2	2	10^{-9}	62
Tree	5	2	10^{-9}	94
Sierpinski tetrahedron	4	3	10^{-9}	109
IFS dragon	2	2	10^{-9}	265
Barnsley fern	4	2	10^{-9}	484
FIF	4	3	10^{-3}	3541
3D Barnsley fern	4	3	10^{-3}	10951

An approximation of the convex hull is computed in an interactive rate for all of the 2D attractors and also for the attractors whose convex hulls have a finite number of vertices.

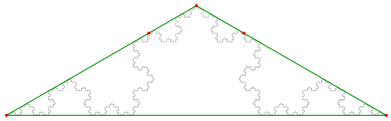
In order to compute an approximate convex hull by Martyn's approach in the same conditions, we implemented it as described in [1]. Table 1 demonstrates the execution time of our algorithm compared to Martyn's for 2D IFS attractors.

One can see that Martyn's approach has strong dependency on the maximal contraction coefficient s_{max} and our algorithm depends more on the number of vertices in the convex hull h.

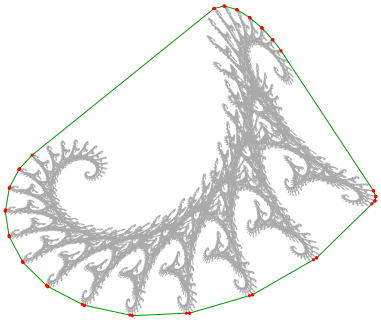
Elimination of the interior points can significantly reduce the number of operations performed, and our optimization therefore provides a qualitative gain in execution time. This is a very important advantage of our method, since more and more tasks in which it is necessary to calculate the convex hull of IFS attractor require computations in real time. Most of design tools to develop fractal structures require interactive rates. Otherwise, such tools simply become unresponsive.



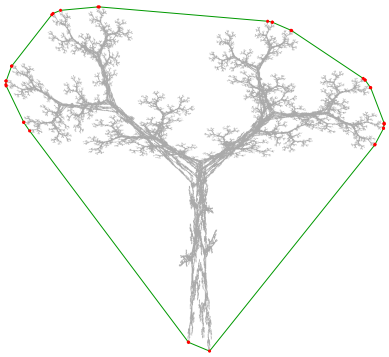
Dragon curve



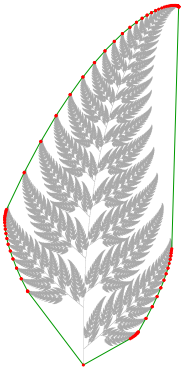
Koch curve



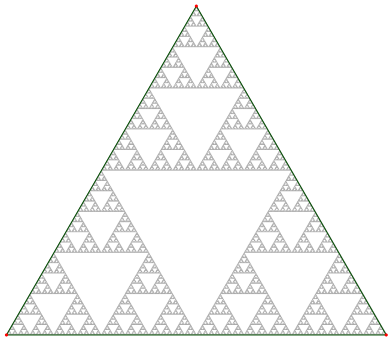
IFS dragon



Tree

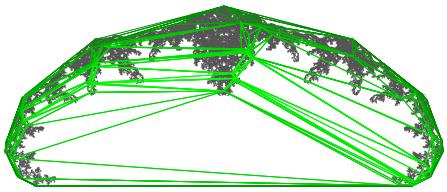


Barnsley fern

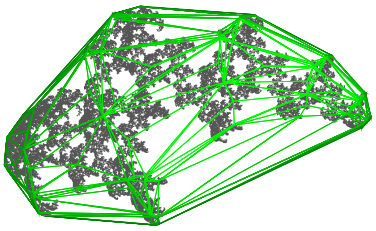


Sierpinski triangle

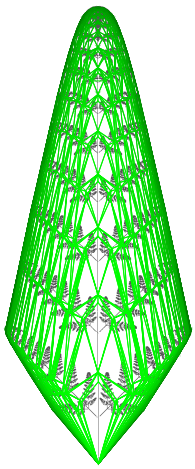
Figure 3: Results for 2D IFS attractors at precision $\varepsilon = 10^{-9}$



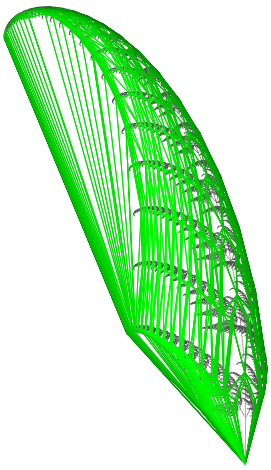
Fractal interpolation function



Fractal interpolation function



3D Barnsley fern



3D Barnsley fern

Figure 4: Results for 3D IFS attractors at precision $\varepsilon = 10^{-2}$

Attractor	N	s_{max}	Martyn's method time, ms	Our algorithm time, ms
Koch curve	4	0.25	125	16
Sierpinski triangle	3	0.5	647	16
Dragon curve	2	0.70711	756	16
Tree	5	0.66762	1131	31
Barnsley fern	4	0.85094	45630	250
IFS dragon	2	0.91944	more than 60 sec	78

Table 1. *The execution time of the algorithms for several IFS attractors with a precision $\varepsilon = 10^{-3}$. Here N is the number of transformations and s_{max} is their maximal contraction coefficient.*

The simplification of the convex hull is another important process. To improve collision test performance it is necessary to verify whether an object intersects with the convex hull of our attractor. This process requires testing all the convex hull faces. The simplified convex hull involves a smaller number of vertices and faces, so verification will take less time. Number of convex hull vertices for $\varepsilon = 10^{-3}$ are presented here:

Attractor	Without simplifying	Simplifying
Tree	30 pts.	25 pts.
IFS dragon	67 pts.	59 pts.
Barnsley fern	186 pts.	101 pts.
3D fern	4577 pts.	359 pts.

In average, the efficiency of the simplification process in 2D is about 46%, but it decreases, when we improve the precision. In 3D, this process is much more efficient - we eliminate about 90% of vertices. However, such efficiency can be achieved only for the attractors whose convex hulls have an infinite number of vertices.

Another important advantage is that we can improve the precision of the obtained approximation by iterating our algorithm further. That is, to improve the precision it is not necessary to recalculate all k iterations, we start with the obtained approximation and continue to iterate the algorithm. This can be useful when it is necessary to compute a sequence of the approximate convex hulls (to obtain different levels of detail for example).

7 Summary

We have presented an iterative algorithm approximating the convex hull of an affine IFS attractor. We have shown how the specific features of the IFS may be exploited to optimize calculations. Our algorithm can be considered as a generalization to 3D and as an optimization of Martyn's algorithm.

For any required accuracy, we construct a sequence of convex hulls in order to eliminate the interior points. To optimize our calculations, we merge the images of the convex hull at each iteration. Moreover, we increase the speed of convergence to the exact convex hull using approximation by ellipses. Performed tests confirm the gain in execution time of our algorithm over that developed by Martyn. Another important advantage is that our algorithm is iterative, i.e. we can improve a precision of the obtained approximation by iterating our algorithm further (without recalculating all previous iterations).

In addition, we have presented a method of simplification which reduces the number of faces in the output convex hull.

References

[1] T. Martyn, *The attractor-wrapping approach to approximating convex hulls of 2D affine IFS attrac-*

tors. Computers & Graphics 33(1) (2009), pp. 104-112.

[2] R. S. Strichartz, Y. Wang, *Geometry of self-affine tiles I*. Indiana University Mathematics Journal 48 (1999), pp. 1–23.

[3] R. Kenyon, J. Li, R. S. Strichartz, Y. Wang, *Geometry of self-affine tiles II*. Indiana University Mathematics Journal 48 (1999), pp. 25–42.

[4] O.S. Lawlor, J.C. Hart, *Bounding recursive procedural models using convex optimization*. Computer Graphics and Applications (2003), pp. 283-292.

[5] J. Duda, *Analysis of the convex hull of the attractor of an IFS*. (http://arxiv.org/ pdf/0710.3863v2), 2008.

[6] T. Martyn, *Realistic rendering 3D IFS fractals in real-time with graphics accelerators*. Computers & Graphics 34(2) (2010), pp. 167-175.

[7] C. Gentil, *Les fractales en synthèse d'images: le modèle IFS*. Thèse de doctorat. Université LYON I, 24 mars 1992. Jury: D. Vandorpe, P. Chenin, J. Mazoyer, J. P. Reveilles, J. Levy Vehel, M. Terrenoire, E. Tosan.

[8] J. C. Hart, T. A. DeFanti, *Efficient anti-aliased rendering of 3D linear fractals.*. Computer & Graphics 25(4) (1991), pp. 91–100.

[9] J. Rice, *Spatial bounding of self-affine iterated function system attractor sets*. Graphics interface GI'96 (1996), pp. 107–15.

[10] T. Martyn, *Tight bounding ball for affine IFS attractor*. Computers & Graphics 27(4) (2003), pp. 535-552.

[11] T. Martyn, *The smallest enclosing disc of an affine IFS fractal*. Fractals. Complex Geometry, Patterns, and Scaling in Nature and Society 17(3) (2009), pp. 269-281.

[12] M. F. Barnsley, *Fractals everywhere*. 2nd ed. Academic Press, Boston; 1993.

[13] S. J. Kim , D. Y. Lee , M. Y. Yang, *Offset triangular mesh using the multiple normal vectors of a vertex*. CAD'04 conference, Thailand; 2004.

[14] D. J. Yoo, *General 3D Offsetting of a Triangular Net Using an Implicit Function and the Distance Fields*. Int. J. Precis. Eng. Manuf., 10(4) (2009), pp. 131–142

[15] T. Chan, *Optimal output-sensitive convex hull algorithms in two and three dimensions*. Discrete & Computational Geometry 16(4) (1996), pp. 361-368.

[16] J. A. Nelder, R. Mead, *A simplex method for function minimization*. Computer Journal 7(4) (1965), pp. 308–313

[17] D. Canright, *Estimating the Spatial Extent of Attractors of Iterated Function Systems*. Computers & Graphics 18(2) (1994), pp. 231-238.

[18] Yu-Xin He, YaLing He, Hua Li, *Fast and accurate determination of the spatial boundary of IFS attractors*. Computers & Graphics 23(4) (1999), pp. 547-553.